Convolutional Neural Networks

Object Detection

Prof. Giuseppe Serra

University of Udine

What are Localization and Detection?

Image Classification

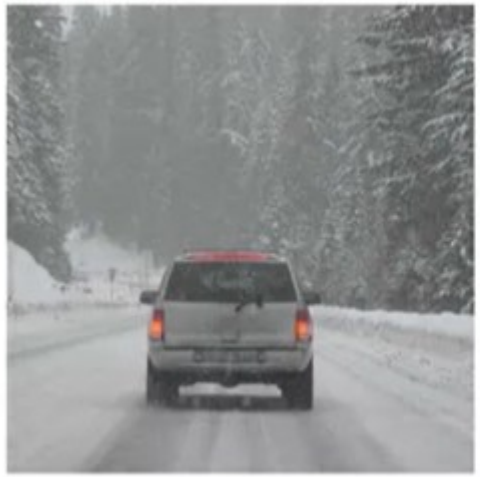
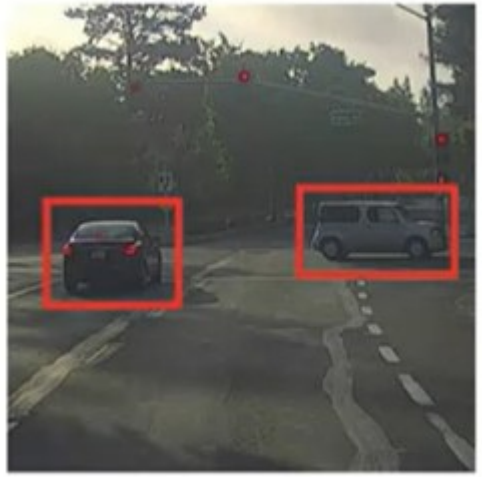


Image Classification
with localization

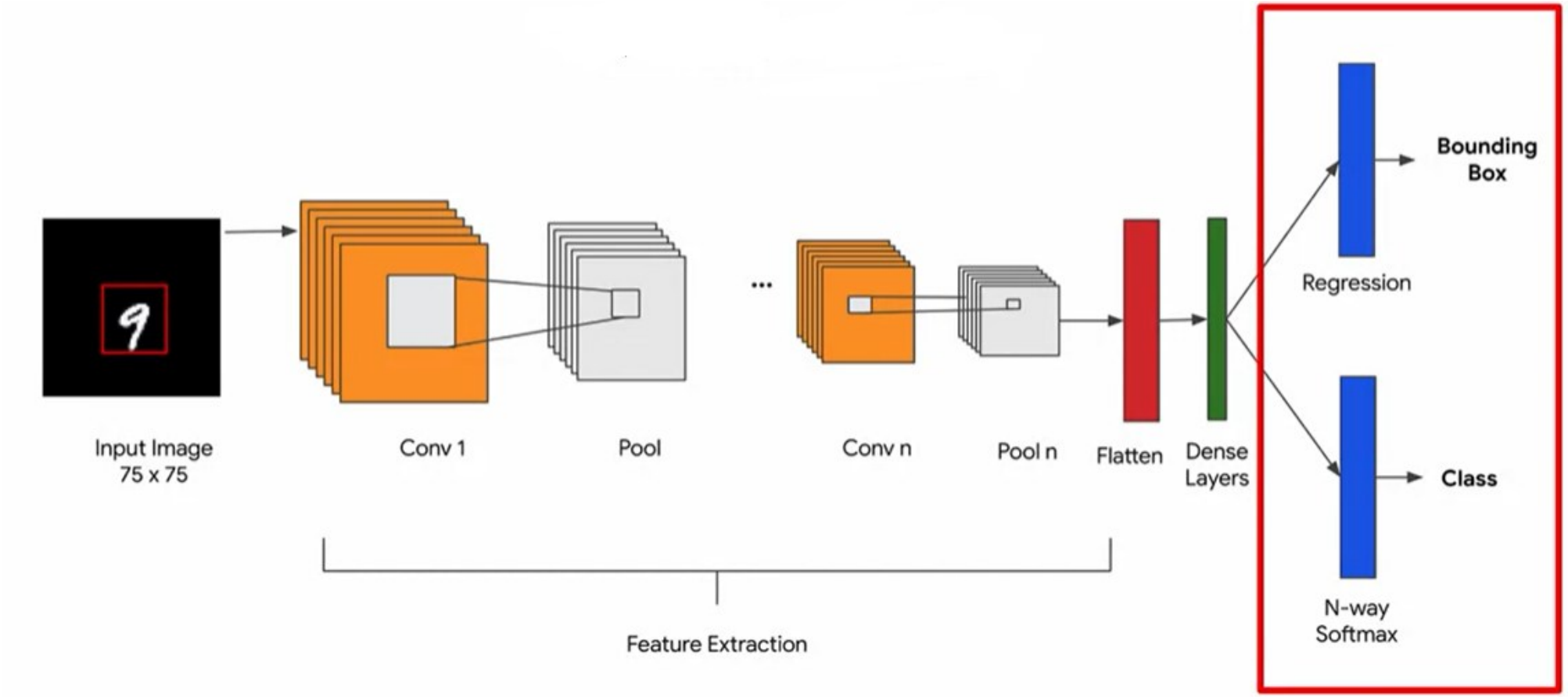
One object

Detection

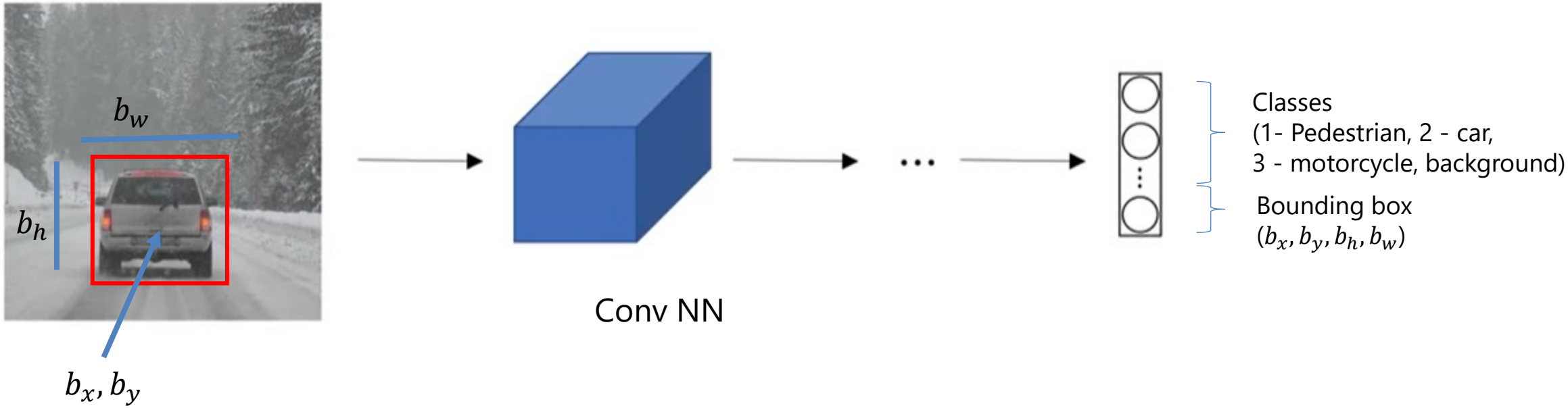


Multiple objects

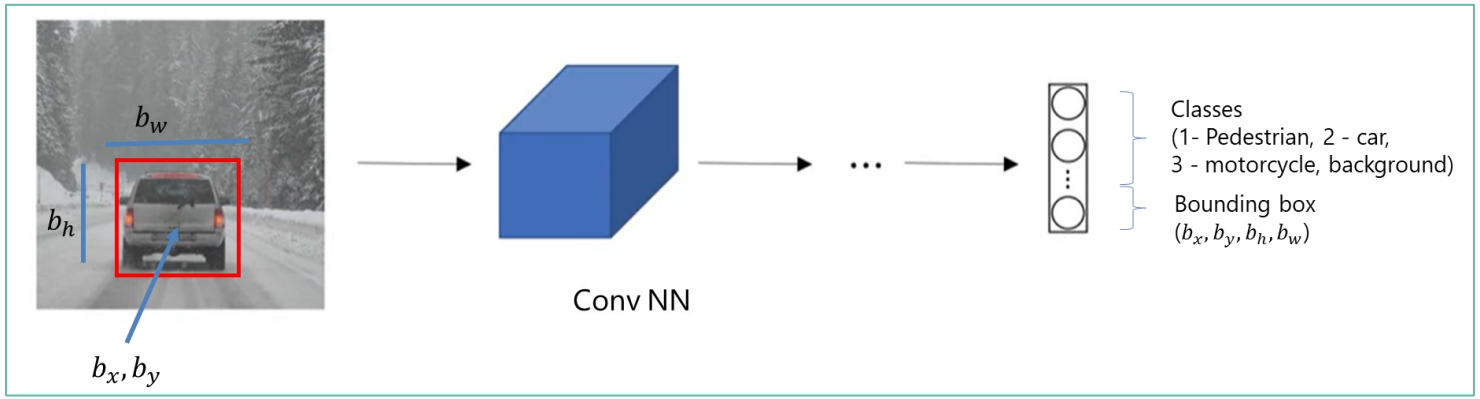
Object Localization



Classification with localization



Classification with localization - Labels



The output: $\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \rightarrow$

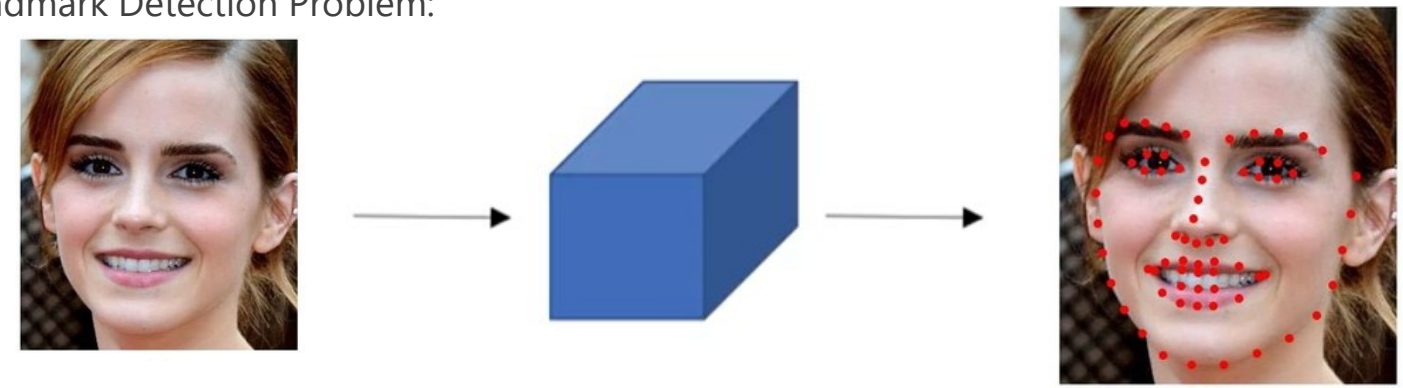
$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$

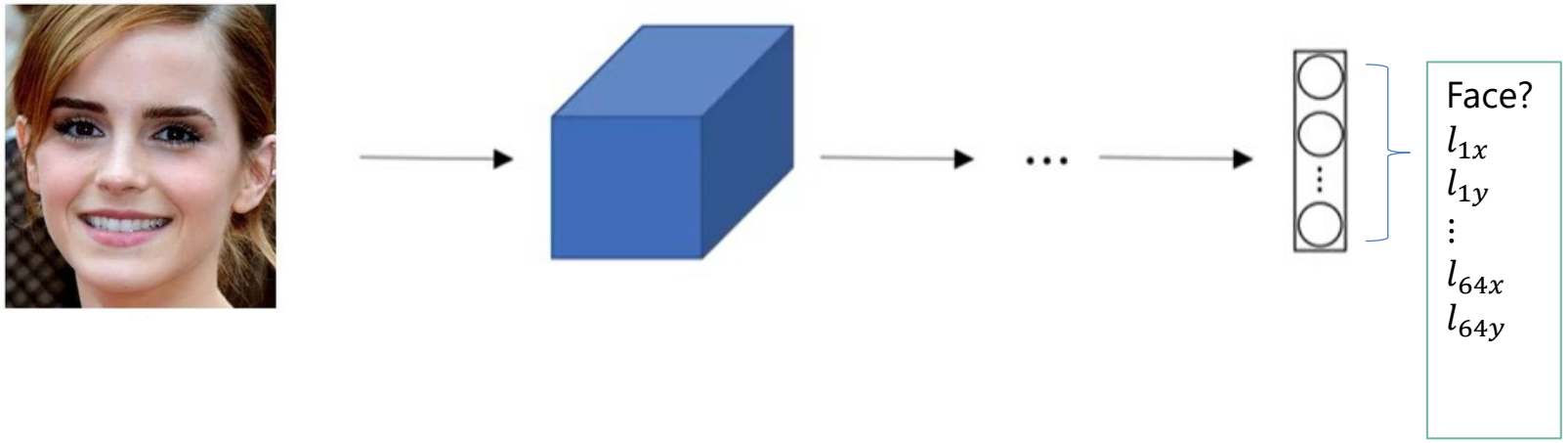
Loss: if there is an object in an image, loss will take care to all the output values, otherwise loss will take car just for the P_c value.

Landmark Detection

Landmark Detection Problem:

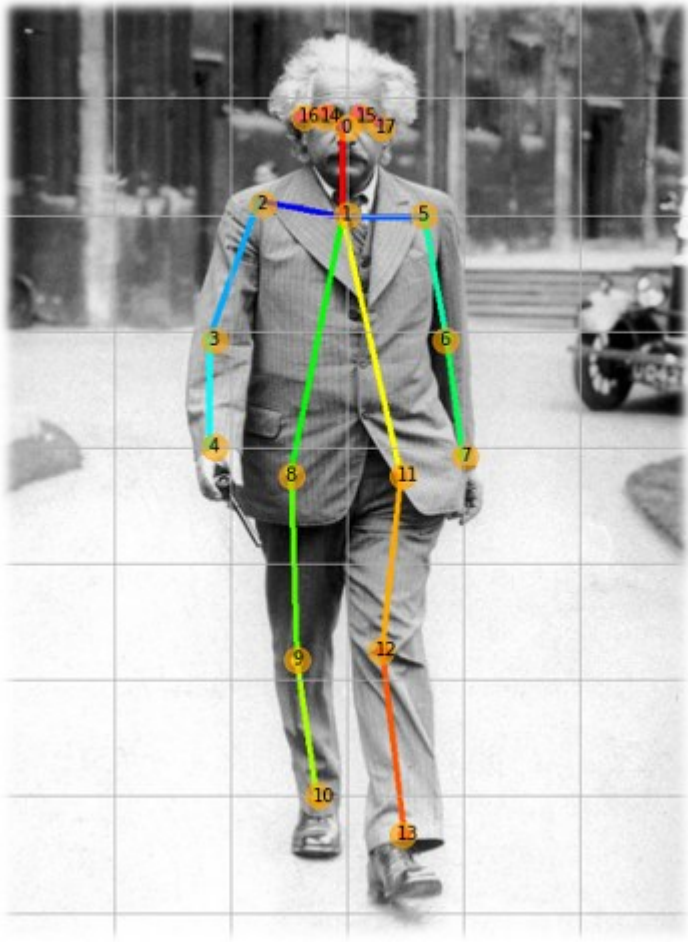


A possible solution:



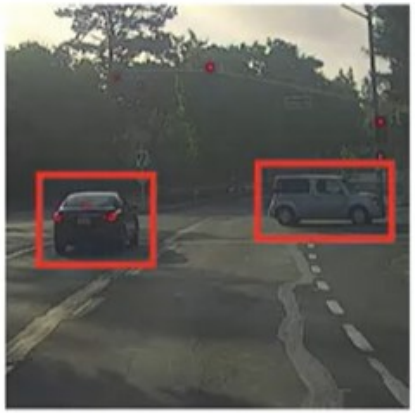
Landmark Detection – Pose Estimation

Same idea can be applied to the full body



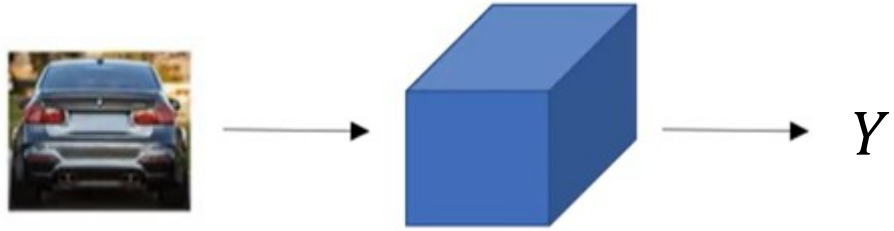
Object Detection

Object Detection



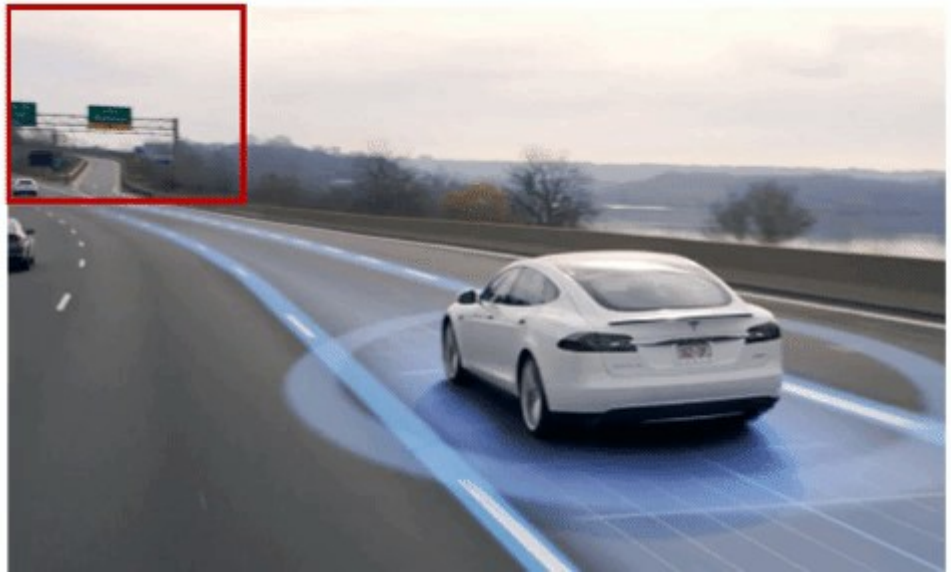
Training set:

x	y
	1
	1
	1
	0
	0



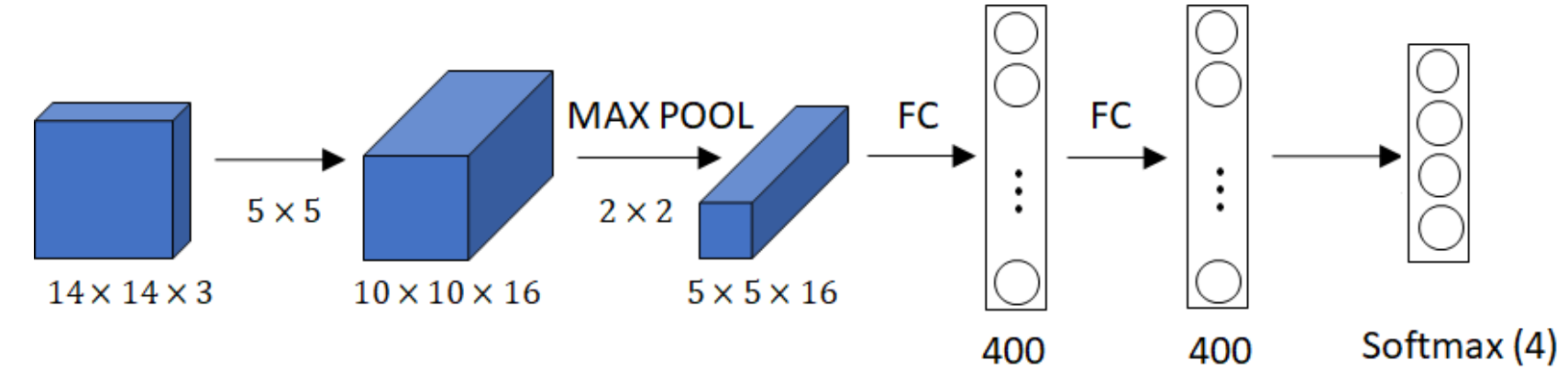
Sliding windows detection

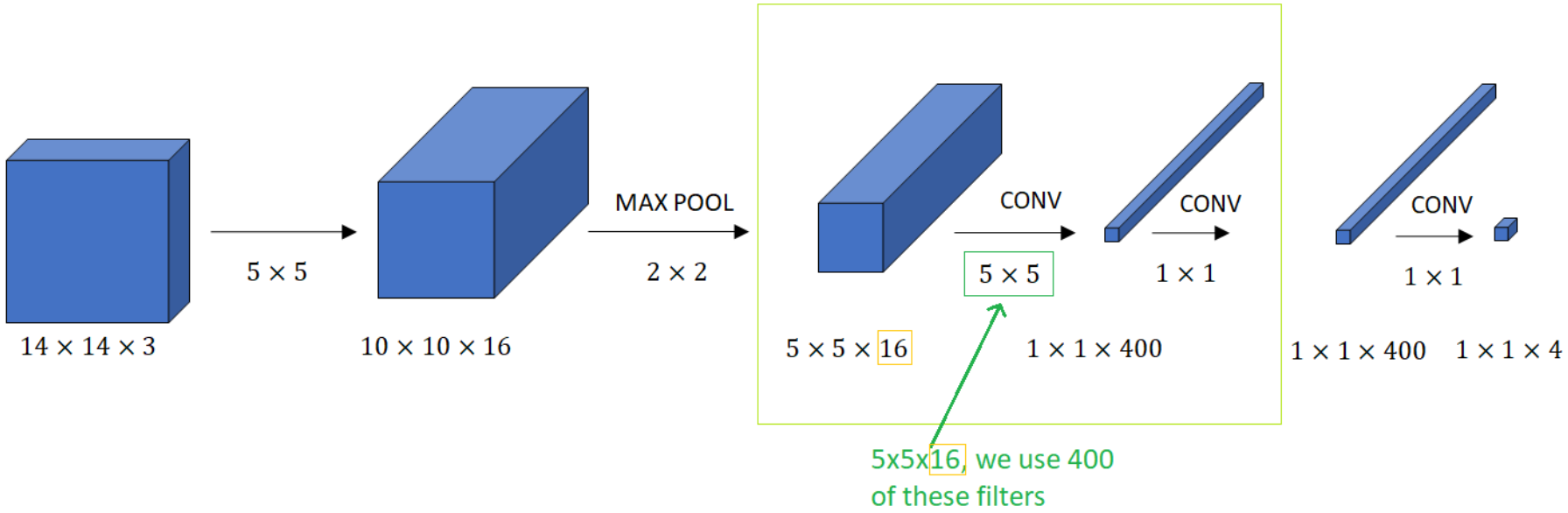
- The idea is to use a sliding window to detect the object
- Each window is used as input of our trained Conv NN



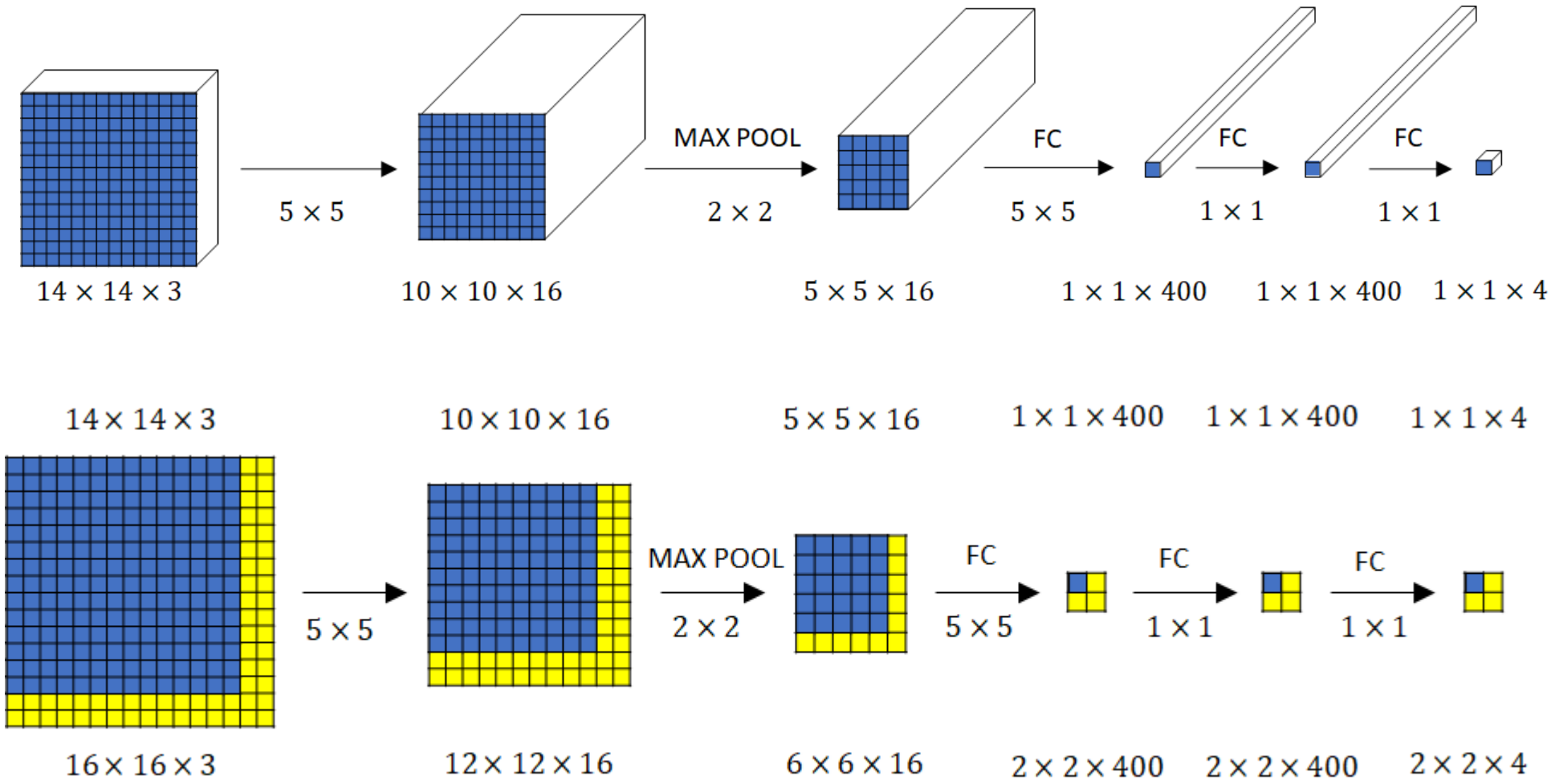
- In order to deal with different scales we just scan the image with windows with different sizes:

Turning FC Layer into Convolutional Layer



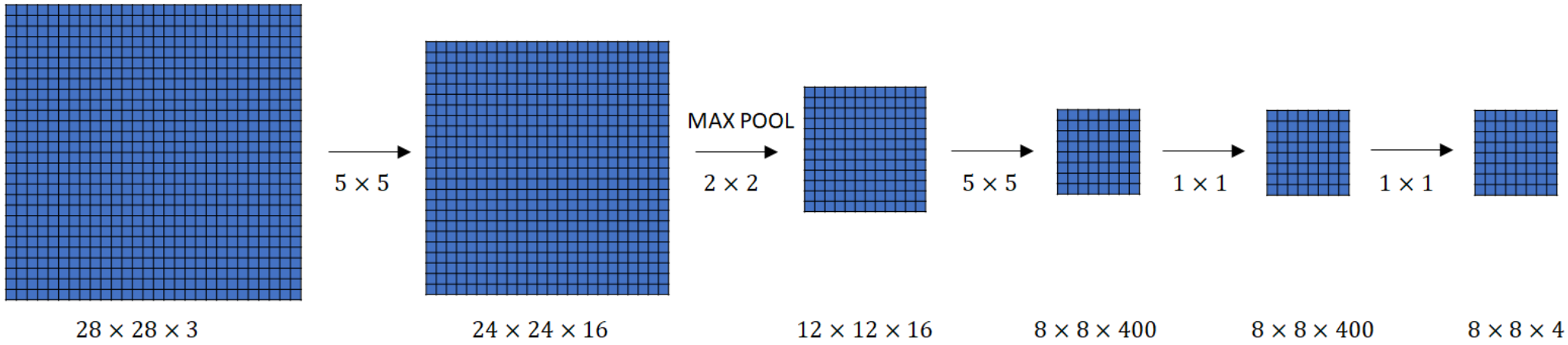


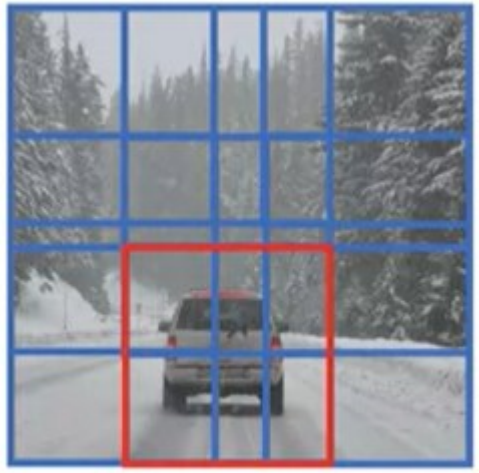
Convolution Implementation of Sliding Windows



[Sermanet et al] OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, 2014

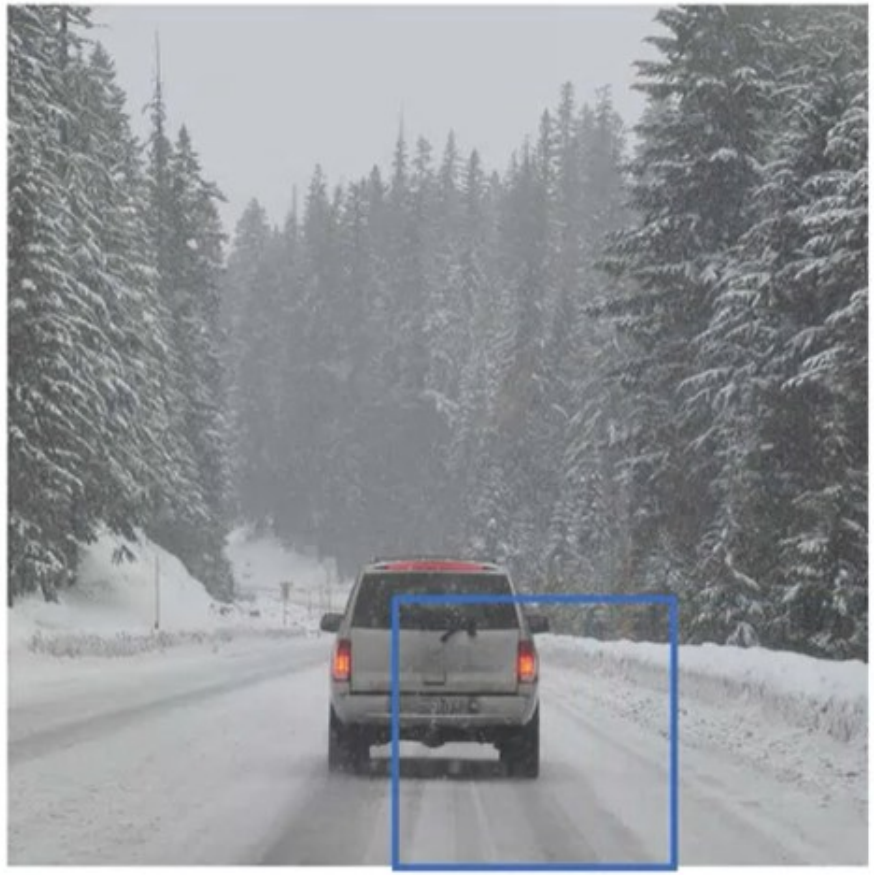
Convolution Implementation of Sliding Windows





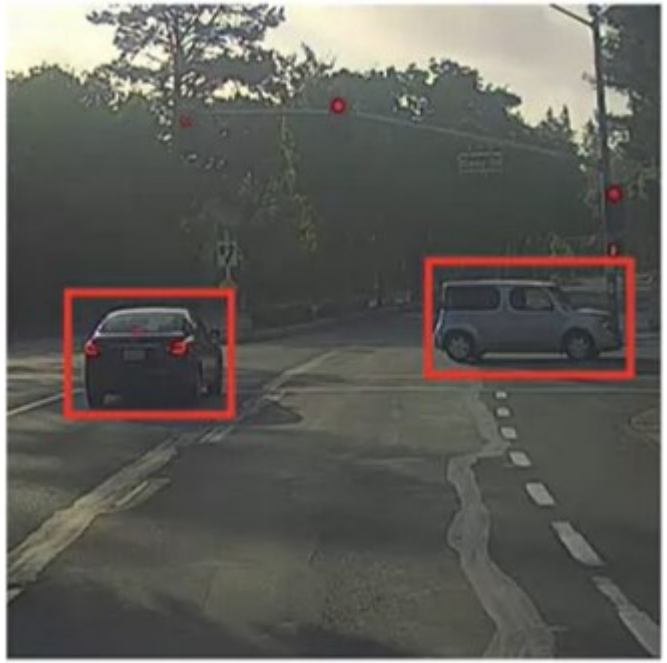
Problem: position of the bounding box is not so accurate

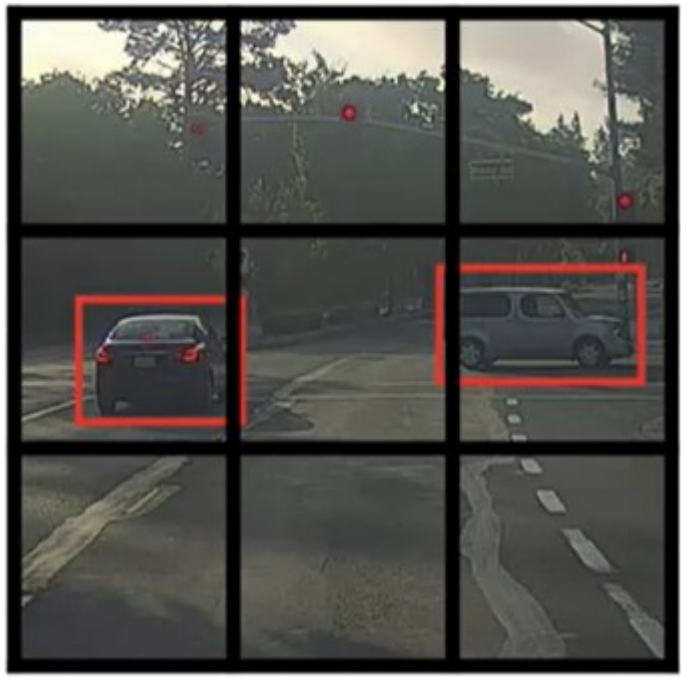
Output accurate bounding boxes



Problem: objects are not always squares…..

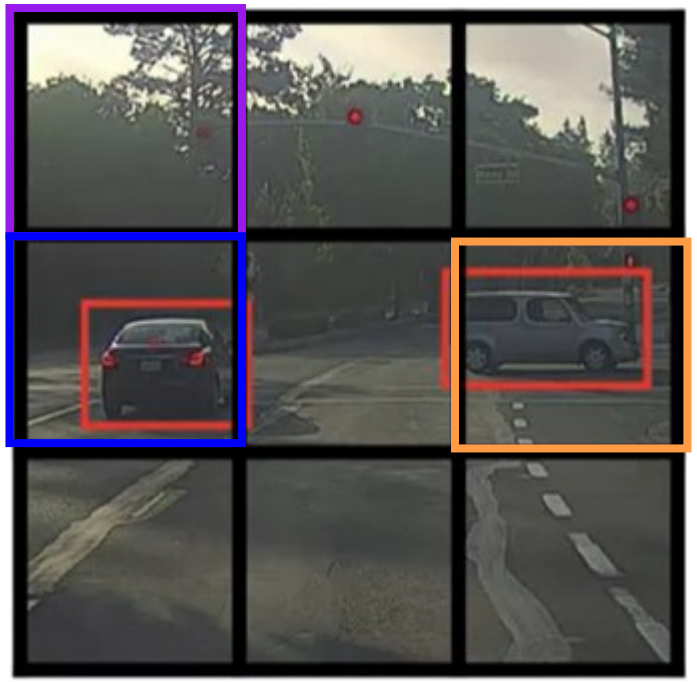
Yolo Algorithm





Here in this example we are using a grid of 3x3, but you can use, for example, a grid of 19x19

Yolo Algorithm



For each grid cell:

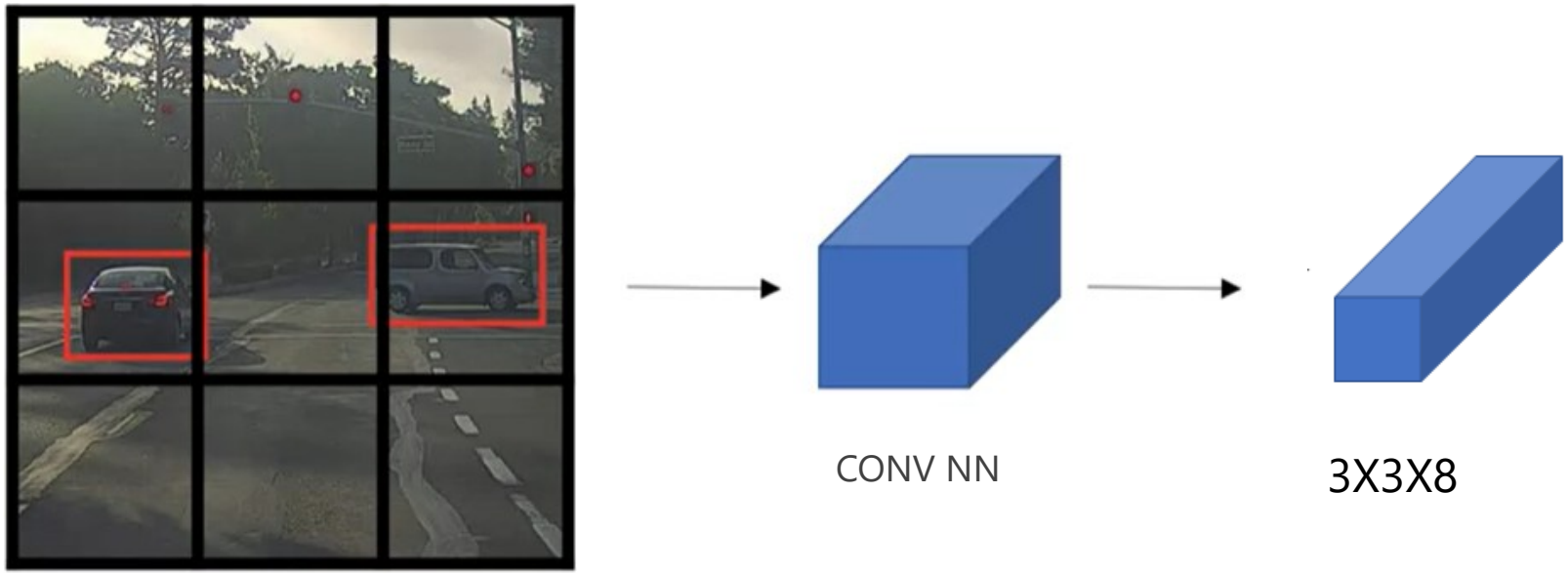
Label : $\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Therefore, for each grid cell the output is 8. Thus, the target output of the image is 3x3x8.

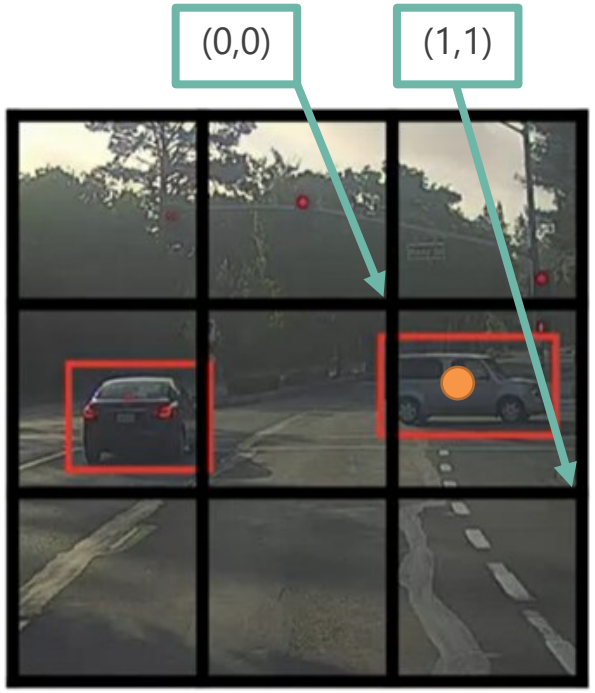
Note each object is associated to a single grid cell. The location of the centroid object determinates the associated grid cell.

[Redmon et al.] You only look once (YOLO)

Yolo Algorithm



Yolo Algorithm



Let's considering the car on the left. The centroid object is colored is orange.

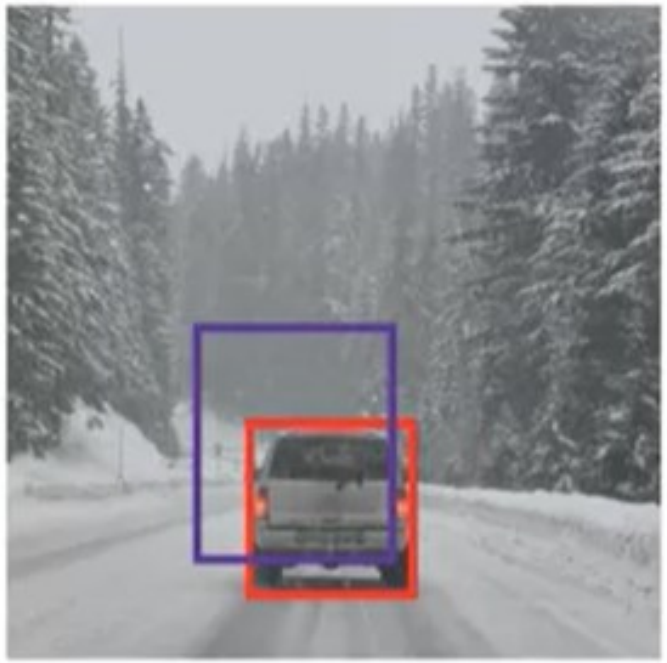
Remember the Label notation : $\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.4 \\ 0.3 \\ 0.9 \\ 0.5 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Centroid should be located in the grid cell, therefore $b_x \in [0,1]$ and $b_y \in [0,1]$

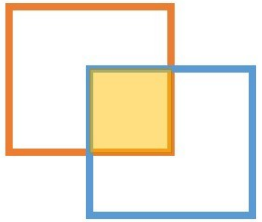
$$b_h = \frac{height}{height\ of\ the\ grid};\ b_w = \frac{width}{width\ of\ the\ grid}$$

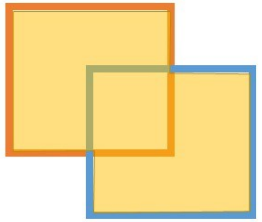
Note b_h and b_w can be greater than 1

Evaluationg Object localization



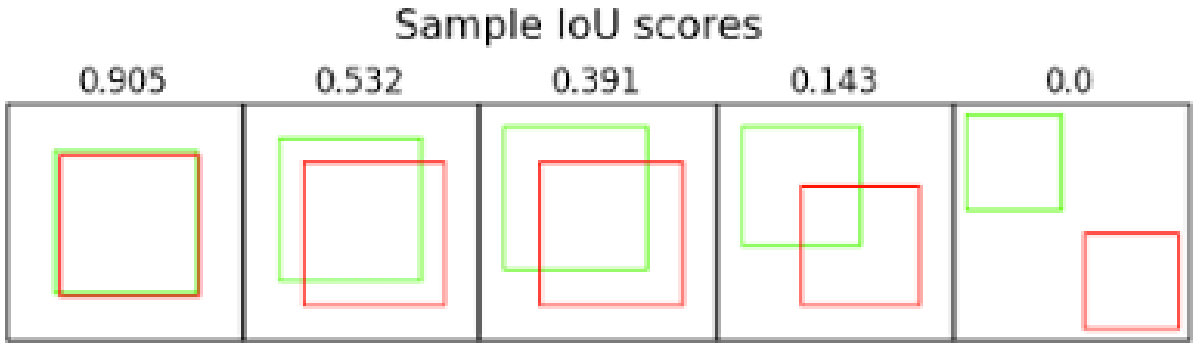
$$Intersection\ over\ Union\ (IoU) = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$





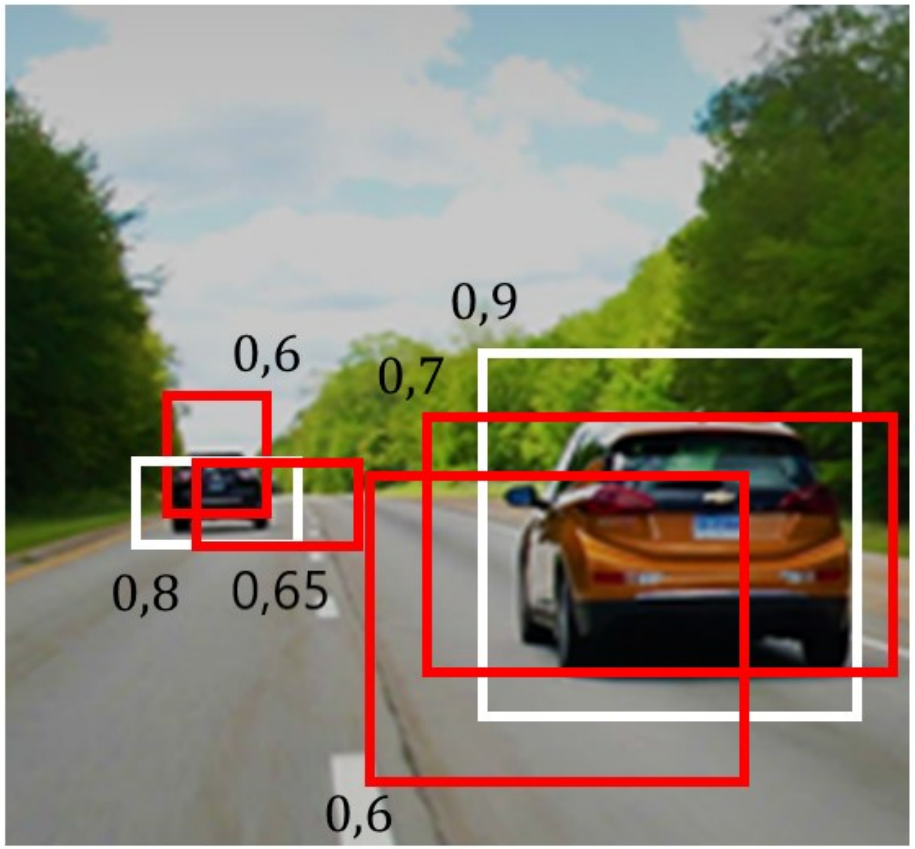
Prediction

Ground-truth



Non-Max Suppression Example

We suppress the no maximum bounding box



Non-Max Suppression Algorithm

Let's considering «car» predictions: $\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Discard all boxes with $P_c < 0.6$

While there are any remaining boxes:

- Pick the box with the largest P_c output
- Discard any remaining box with IoU>0.5 with the box output in the previous step

Note: If you have a multi-object detection do it for each of them independently

Let's considering «car» predictions: $\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Discard all boxes with $P_c < 0.6$

While there are any remaining boxes:

- Pick the box with the largest P_c output
- Discard any remaining box with IoU>0.5 with the box output in the previous step

Image Segmenation

Image Segmentation

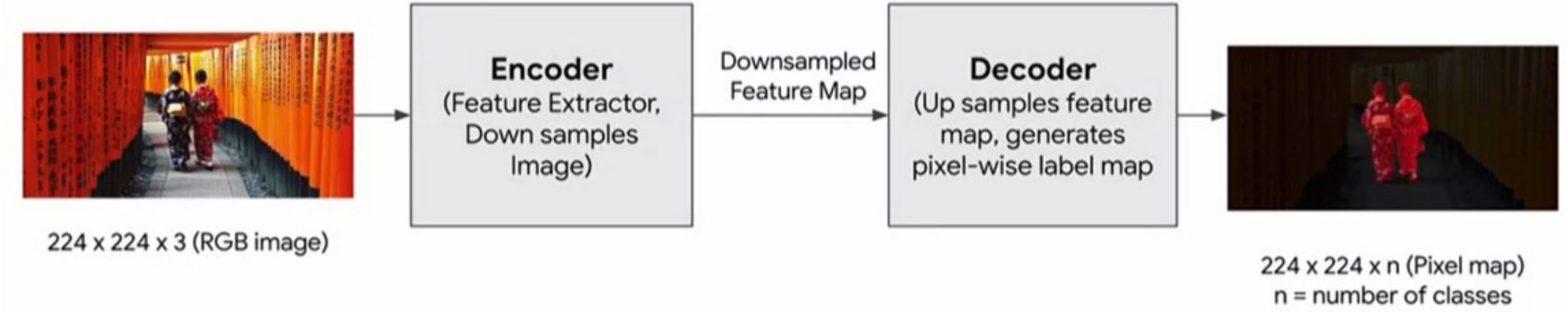
Semantic Segmentation

Instance Segmentation

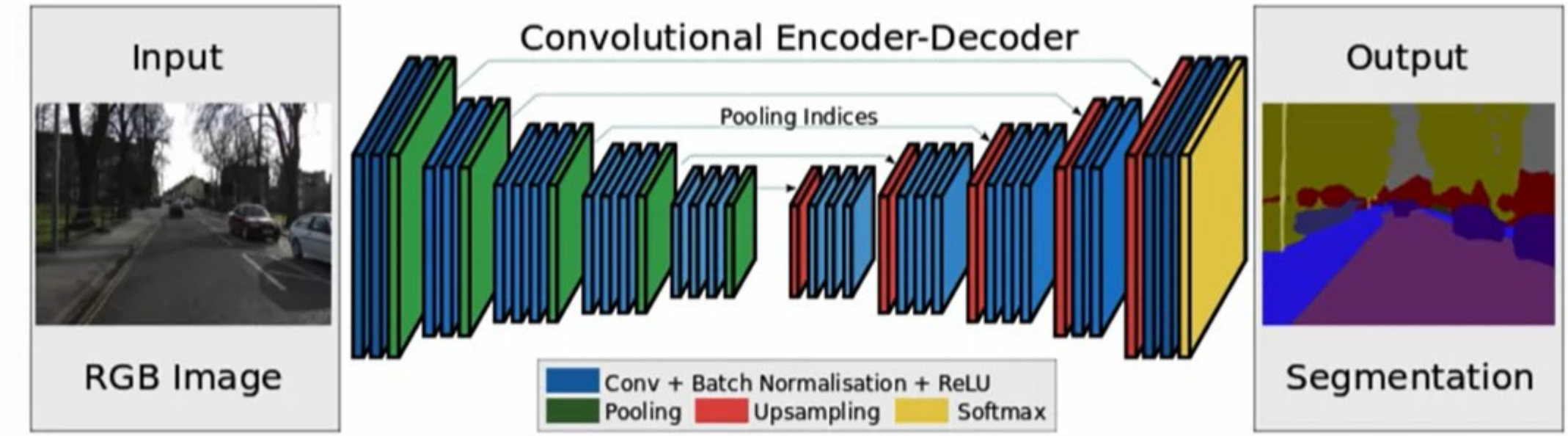
Training

Classes indices = [0 = Background 1 = People]

Image Segmentation Basic Architecture



SegNet


Input
RGB Image
Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax
Output
Segmentation

Simple Scaling – UpSampling 2D

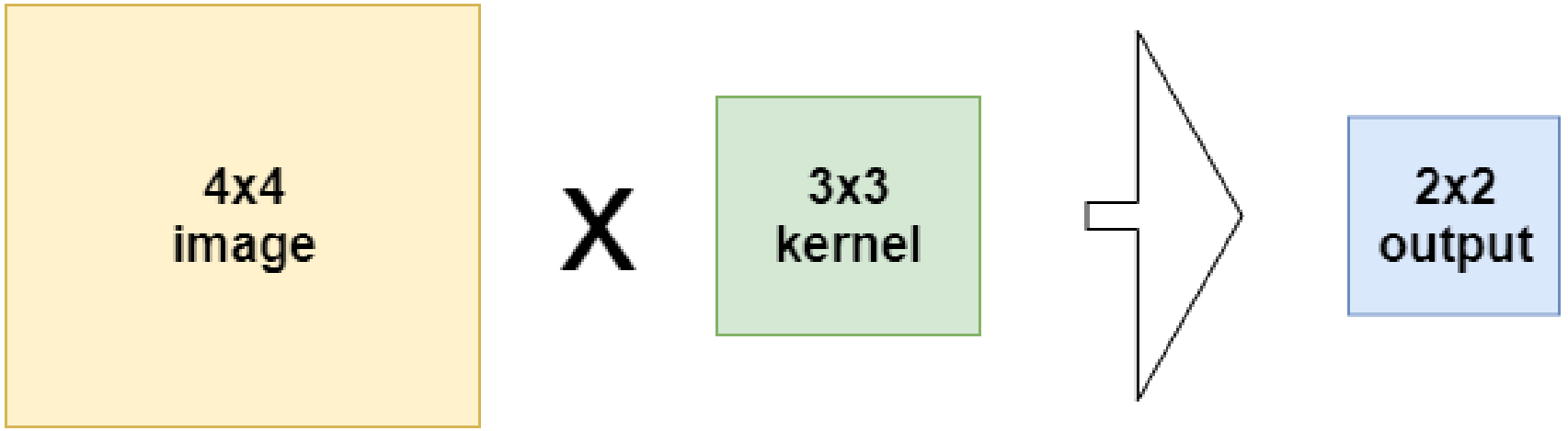
- Upsampling2D scales up the image
- Two Types of scaling:
  - Nearest
    - Copies value from nearest pixel.
  - Bilinear
    - linear interpolation from nearby pixels.

Transposed Convolutions

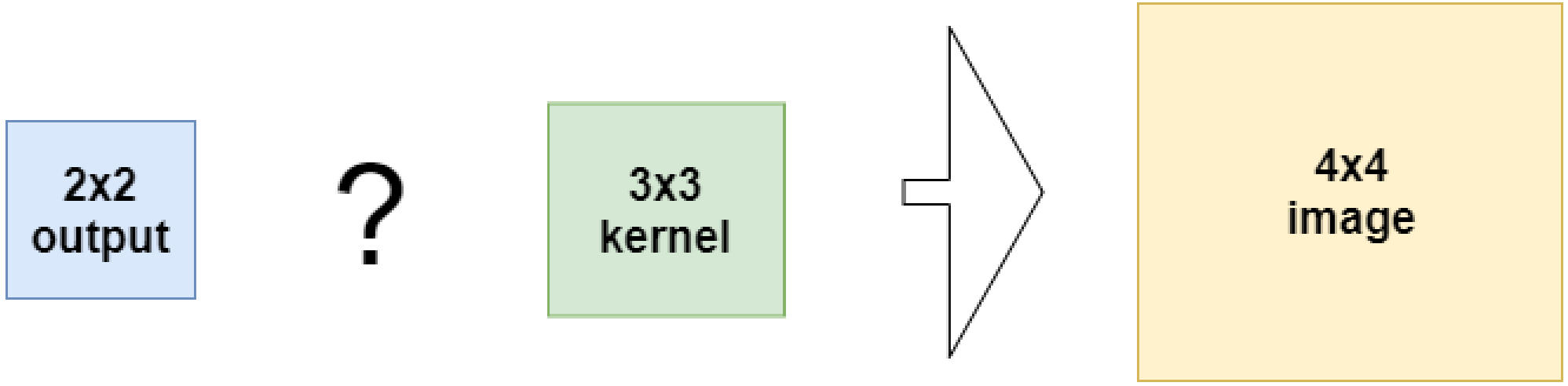
Transposed Convolution layer is a variation of classical Convolution layer, which can be used for upsampling images.

To understand this variation is important to revised the convolution layer idea.

Normal convolution:



Transposed Convolutions: the goal



Convolution Matrix

Let's now see **how we can represent the normal convolution by means of a Convolution Matrix**.

Suppose that we're performing convolutions with a **2×2 kernel** on a **3×3 input image**, like this:

1	2	3
6	5	3
1	4	1

3x3 Input

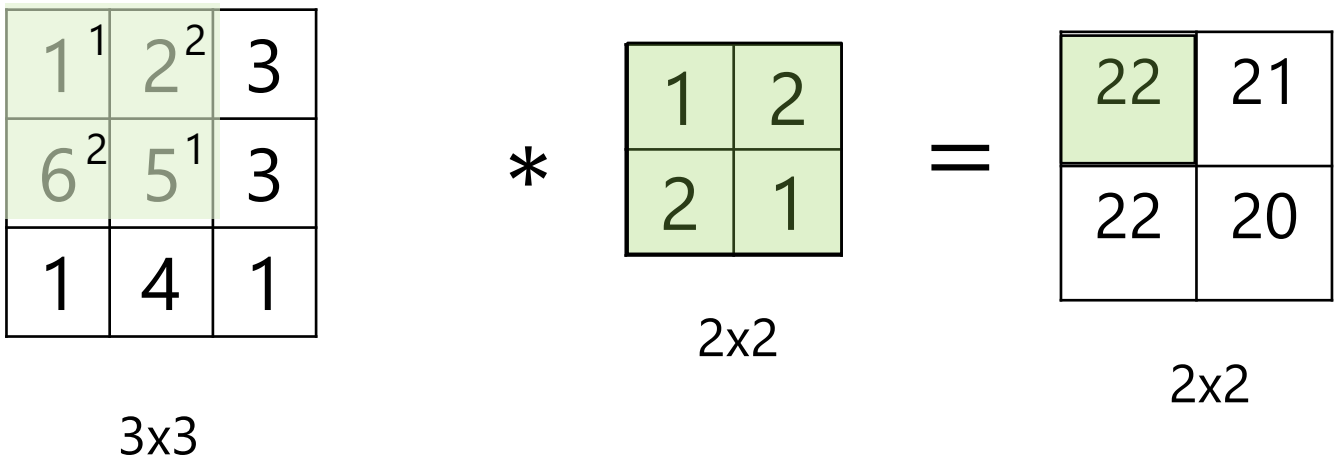
1	2
2	1

2x2 Kernel

With our understanding of how regular convolutions work, it's not surprising to find that we'll end up with a 2×2 output or feature map:

22	21
22	20

Details



Convolution Matrix

1^{1}	2^{2}	3
6^{2}	5^{1}	3
1	4	1

We can also represent this operation as a Convolution Matrix.

It's a matrix which demonstrates all positions of the kernel on the original image, like this:

1	2	0	2	1	0	0	0	0
0	1	2	0	2	1	0	0	0
0	0	0	1	2	0	2	1	0
0	0	0	0	1	2	0	2	1

The {1, 2, 0} at the first row of the convolution matrix therefore represents the effect of the convolution at the first row of the input image. The {2, 1, 0} represents the effect of the convolution at the second row of the input image. Since at this point, the convolution is not applied in either the 3rd column or the 3rd row, either the third column value of the first and second row *and* all the third row values are 0.

Convolution Matrix

We can also reshape the input image into a (9x1) feature vector:

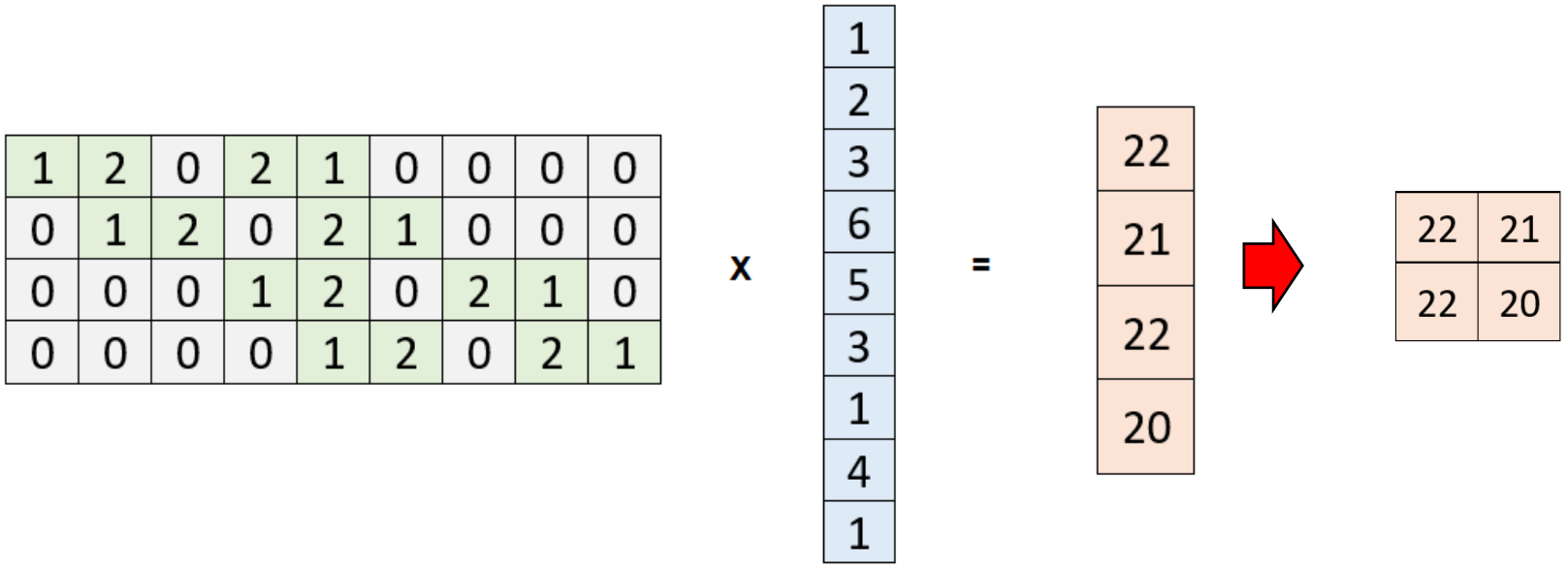
1	2	3
6	5	3
1	4	1

=

1
2
3
6
5
3
1
4
1

Convolution Matrix

The fun thing is that we can multiply this (9×1) matrix with the (4×9) convolution matrix and hence achieve a (4×9) x (9×1) = (4×1) output:



From output back to the input: the Transposed Convolution

Now suppose that this is your input:

1	2
2	4

While this is your desired output:

1	4	4
4	13	10
4	10	4

From output back to the input: the Transposed Convolution

We find why **we call the type of convolution *transposed***, as we can also represent this by means of the convolution matrix – although **not the original one, but its transpose**:

1	0	0	0
2	1	0	0
0	2	0	0
2	0	1	0
1	2	2	1
0	1	0	2
0	0	2	0
0	0	1	2
0	0	0	1

From output back to the input: the Transposed Convolution

Now we now have a (9×4) matrix and a (4×1) matrix, which we can multiply to arrive at a (9×1) matrix or, when broken apart, the (3×3) matrix we were looking for!

1	0	0	0
2	1	0	0
0	2	0	0
2	0	1	0
1	2	2	1
0	1	0	2
0	0	2	0
0	0	1	2
0	0	0	1

x

1
2
2
4

=

1
4
4
4
13
10
4
10
4

=

1	4	4
4	13	10
4	10	4